Chapter VIII
The Mot+Visual Language for Knowledge-Based Instructional Design

Gilbert Paquette
Affiliation, Country

Michel Léonard
Affiliation, Country

Karin Lundgren-Cayrol
Affiliation, Country

ABSTRACT

his chapter states and explains that a Learning Design is the result of a knowledge engineering process where knowledge and competencies, learning design, media and delivery models are constructed in an integrated framework. Consequently, we present our MOT+ general graphical language and editor that help construct structured interrelated visual models. The MOT+LD editor is the newly added specialization of this editor for learning designs, producing IMS-LD compliant Units of Learning. The MOT+OWL editor is another specialization of the general visual language for knowledge and competency models based on the OWL specification. We situate both models within our taxonomy of knowledge models respectively as a multi-actor collaborative process and a domain theory. The association between these "content" models and learning design components is seen as the essential task in an instructional design methodology, to guide the construction of high quality learning environments.

INTRODUCTION

Building high quality learning designs is a very important and demanding task. It is also a difficult task that we started to address already a decade ago by progressively building an instructional engineering method (Paquette et al., 1994, 2005a; Paquette, 2003), a delivery system (Paquette et al., 2005b) and a graphical knowledge modeling editor (Paquette, 1996, 2002).

In this on-going work and for the present discussion, the point of view is taken that a learning design is the result of a knowledge engineering process, where knowledge and competencies,

learning design, media and delivery models are constructed in an integrated framework. In the first section of this chapter, we present the MISA[1] instructional design method based on these four models and their relationships to each other. The second section presents the MOT (Modeling with Object Types) visual language and the specialized editing tools that have been used in numerous applications. We summarize the theoretical basis of the language, its syntax and semantic, moreover examples within the MISA instructional design method will be presented.

The third and fourth sections address the standardization issues and how the MOT+ software is adapted to provide visual aid to designers building knowledge and/or pedagogical models. The third section focuses on the learning design models, the IMS-LD specification and the specialized MOT+LD editor that helps designers build IMS-LD compliant and interoperable units of learning. The fourth section presents the Ontology Web Language (OWL) and the specialized MOT+OWL visual editor. We use it to represent domain knowledge models and target competency that can be used to plan, support staff roles and evaluate the quality of learning designs. In the fifth section we discuss the association between LD models and OWL models to support what we believe is the central task for knowledge-based instructional design aiming to support learning environments within the Semantic Web.

Finally, the concluding section will summarize the properties of representation languages that we have found most useful while designing and using the various specializations of the MOT+ software through its evolution from a general knowledge modeling tool to a standardized tool at the heart of the instructional design methodology.

INSTRUCTIONAL DESIGN BASED ON VISUAL MODELING

In this section, we present a synthesis of the main MISA 4.0 Instructional Engineering Method components and concepts. A knowledge modeling approach using the MOT editor was used to define the Instructional Engineering method itself, its concepts, processes and principles. And thus, this method can also be seen as a visual modeling application.

This R&D initiative, started in 1992, has led to the MISA 4.0 version (Paquette, 2001a, 2002a) and to its support tool, called ADISA[2] (Paquette et al., 2001). The editor MOT+ is embedded in the ADISA system and accessible through a Web browser from workstations linked to the Internet. It can also be used without ADISA together with forms provided by the MISA documentation. Since 2001, the method has been adapted to the huge standardization work that has occurred in the e-Learning sector; we will address this aspect in later sections of this chapter.

Overview of the Method

The MISA Learning Engineering process produces specifications of learning environment grouped in documents called Documentation Elements (DE). Table 1 presents these DEs.

Each DE results from tasks distributed into six phases. Within phase 2, 3, 4 and 6, these DE can also be viewed according to four axes or dimensions of an e-Learning environment: Knowledge, Pedagogy, Media and Delivery. Presently, MISA 4.0 comprises 35 basic sub-tasks, each producing one DE, numbered, as shown in table 1, from 100 to 640. The first digit denotes the phase,

Table 1. MISA 4.0 documentation elements—phases and axes

Phase 1: Definition	100 Organization's Training System 102 Training Objectives 104 Learners' properties 106 Present Situation 108 Reference Documents			
	Knowledge Axis	Pedagogy Axis	Media Axis	Delivery Axis
Phase 2: Initial solution	210 Knowledge Model Orientation Principles ***212 Knowledge Model*** 214 Target Competencies	220 Instructional Principles ***222 Learning Event Network*** 224 Learning Unit Properties	230 Media Principles	240 Delivery Principles 242 Cost-Benefit Analysis
Phase 3: LE architecture	***310 Learning Unit Content***	**320 Learning Scenarios** 322 Activity Properties	330 Development Infrastructure	340 Delivery Planning
Phase 4: LE detailed Design	410 Learning Resource Content	420 Learning Resource Properties	430 Learning Resource List ***432 Learning Resource Models*** 434 Media Elements 436 Source Doc.	***440 Delivery Models*** 442 Actors and their resources 444 Tools and Telecommunication 446 Delivery Services
Phase 5: Validation	540 Test Planning 542 Revision Decision Log			
Phase 6: Delivery Plan	610 Knowledge/Competency Management	620 Actors and Group Management	630 Learning System/Resource Management	640 Maintenance/Quality Management

the second, the axis, and the third, the sequence number within the axis. A DE is either a visual model, identified in bold italic in table 1, or a text-based form describing guidelines for a model or properties of objects in the model.

A Problem Solving Approach in 6 Phases

MISA proposes a problem solving approach. Each MISA phase is subdivided into a number of steps where parts of a learning environment or system are constructed. These phases are sequential, but spiral, with frequent returns to modify the result or previous tasks:

- **Phase 1:** Designers build a description of the training problem, its context and constraints. The general goal that the solution must fulfill and the main characteristics of the target population are the most important aspects to address at this point.
- **Phase 2:** Designers define a preliminary training solution, centered on a knowledge model for the learning domain. Prerequisite and target competencies are associated to the most important knowledge entities in the model. In this phase, designers also build a first pedagogical visual model called "the learning event network" grouping the main modules or learning units, their sequencing and the resources needed to perform them or to be produced by learners and facilitators.
- **Phase 3:** Designers construct a detailed learning design and specify the infrastructure necessary. Visual learning scenarios are built for each learning unit defined in phase 2, describing the learning and facilitating activities, the actors that perform them and the resources needed or produced by these actors. At the same time, a sub-model of the phase 2 knowledge model is associated with each learning unit thus defining "the learning unit content." According to the evolution of the design, media and delivery principles are refined to prepare the next phase.
- **Phase 4:** Centered on the learning resources and delivery models and the properties of objects in these models several professionals may work together: content experts, instructional designers and media designers. Another important concurrent task is the

description of the properties of resources in learning scenarios and the association of a sub-model of the knowledge model to provide a specification of the "learning resource content."

- **Phase 5:** The project manager plans the validation of the learning environment and produces a list of possible revisions and decisions about how to improve the specifications created in the previous phases.
- **Phase 6:** Designers and project manager prepare elements necessary to the delivery of the learning environment. It produces a synthetic and global description of the learning environment for its maintenance and quality management by various actors.

A Visual Modeling Approach

In each of phases 2, 3 and 4, MISA also proposes the development of the learning environment along four axes: knowledge and competency (content model), instructional, resources and delivery. The central product of each axis is one or more visual models.

The *knowledge model* centers on a graphical representation of the learning environment content domain. In this model, the domain's facts, concepts, procedures and principles are displayed and interrelated with precise links. Then target and prerequisite competencies are linked to knowledge elements in the model, thus identifying prerequisites and learning objectives for the Pedagogical Model. Subsequently, knowledge units and competencies are also associated to learning units and to the resources present in the learning units' scenario models.

The *instructional model* is essentially a visual network of learning events and units, to which knowledge and target competencies are associated. Each learning unit is also described by a visual learning scenario specifying learning and support activities linked to resources in the environment. Resources holding content (as opposed to tools and services) are associated with a subset in the knowledge model.

The *learning resource models* are useful to describe materials (or learning objects) to be adapted and produced, their media components, source documents and presentation principles as well as other properties aimed at graphical designers and learning material producers.

Finally, *delivery models* are produced to show how and where actors use or provide learning materials and resources such as tools, communication means, services and locations, used in the learning environment. Each Delivery Model is a multi-user workflow, where actors use or produce resources, while assuming different roles. These processes address organizational issues, such as group organization, staff assignments, technical help, resource delivery, and so on, which must be prepared to ensure smooth deployment of a network-based or a distance learning environment.

Each and every one of these models is built using the MOT+ knowledge representation technique and tool (Paquette, 1999, 2002b). Graphical visual models are the basic DE in each axis, the backbone of the MISA method. Most of the other tasks, in MISA, describe properties of objects in these models (e.g., competencies, learning units, resources, roles) as well as their relationships.

MOT+: A GENERIC VISUAL LANGUAGE AND TOOL

When designers start building a learning environment, two basic questions arise: "Which knowledge must be acquired, what are the target competencies or educational objectives for that knowledge?" and "How should the activities and the resources be organized to best achieve knowledge and competency acquisition?" To help designers solve this type of questions, we have developed a graphical knowledge modeling method and tools, thus visualizing activity sequences, actors and tools. In this section, we present the

MOT modeling language that serves that purpose and the MOT+ visual modeling editor.

The graphic or visual representation formalism that we present here (Paquette, 1996; Paquette, 2002) has been tested for the past 10 years in a vast array of modeling applications and in many various contexts. It is used by trainers for corporate training, and designers or professors use it to prepare university courses or to propose modeling exercises to their students. It has served to model processes for the implementation of a computer-supported high school, or to model instructional methods or research projects processes.

Basis for a Graphical Knowledge Representation Language

It is often said that a picture is worth a thousand words. That is true of sketches, diagrams, and graphs used in various fields of knowledge. *Conceptual maps* are widely used in education to represent and clarify complex relationships between concepts. *Flowcharts* are graphical representations of procedural knowledge or algorithms. *Decision trees* are another form of representation used in various fields, particularly in decision-making expert systems.

All these representation methods are useful at an informal level, as thinking aids and tools for the communication of ideas, but they also have their limitations. One is the imprecise meaning of the links in a model. Another issue is the ambiguity around the type of entities or symbol system that is used. Objects, actions on objects and statements of properties about them are all mixed-up, which make graph interpretation a fuzzy and risky business. Another difficulty is to combine more than one representation in the same model. For example, concepts used in procedural flowcharts as entry, intermediate or terminal objects could be given a more precise meaning by developing them in conceptual sub-models of the procedure. The same is true of procedures present in conceptual models that could be developed as procedural sub-models described by flowcharts, combined or not with decision trees.

In software engineering, many graphic representation formalisms have been or are used such as Entity-Relationship models (Chen, 1976), Conceptual Graphs (Sowa, 1964), the Object Modelling Technique (OMT) (Rumbaugh, Blaha, Premerlani, Eddy& Lorensen, 1991), KADS (Schreiber, Wielinga & Breuker, 1993) or the Unified Modeling Language (UML) (Booch, Jacobson & Rumbaugh, 1999). These representation systems have been built for the analysis and architectural design of complex information systems. The most recent ones require the use of up to eight different kinds of model and links, which rapidly become hard to follow without considerable expertise.

Our initial goals were different. We needed a graphic representation system that was both simple enough to be used by educational specialists, such as teachers, professors and tutors, who are not, in general, computer scientists, still general and powerful enough to represent the components and their relationships of computer-based educational environments.

There is a consensus in educational science to distinguish four basic types of knowledge entities (facts, concepts, procedure and principles), despite some diversity in terminology and definitions. See for example, the work of Merrill (1994), Romiszowski (1991), Tennyson and Rash (1988), and West , Farmer and Wolf (1991). This categorization is retained as the basis for the MOT graphic representation language.

All four types of knowledge are also considered in the framework of schema theory. The concept of schema is the essential idea behind the shift from behaviourism to cognitivism, the now dominant theory in psychology and other cognitive sciences, based on the pioneering ideas of Inhelder and Piaget (1958) as well as Bruner (1973). In the early seventies, Newell and Simon (1972) developed, on the same basis, a rule-based representation of the human problem solving procedural activity, while Minski (1975) defined the concept of

"frame" as the essential element to understand perception, and also to reconcile the declarative and procedural views of knowledge.

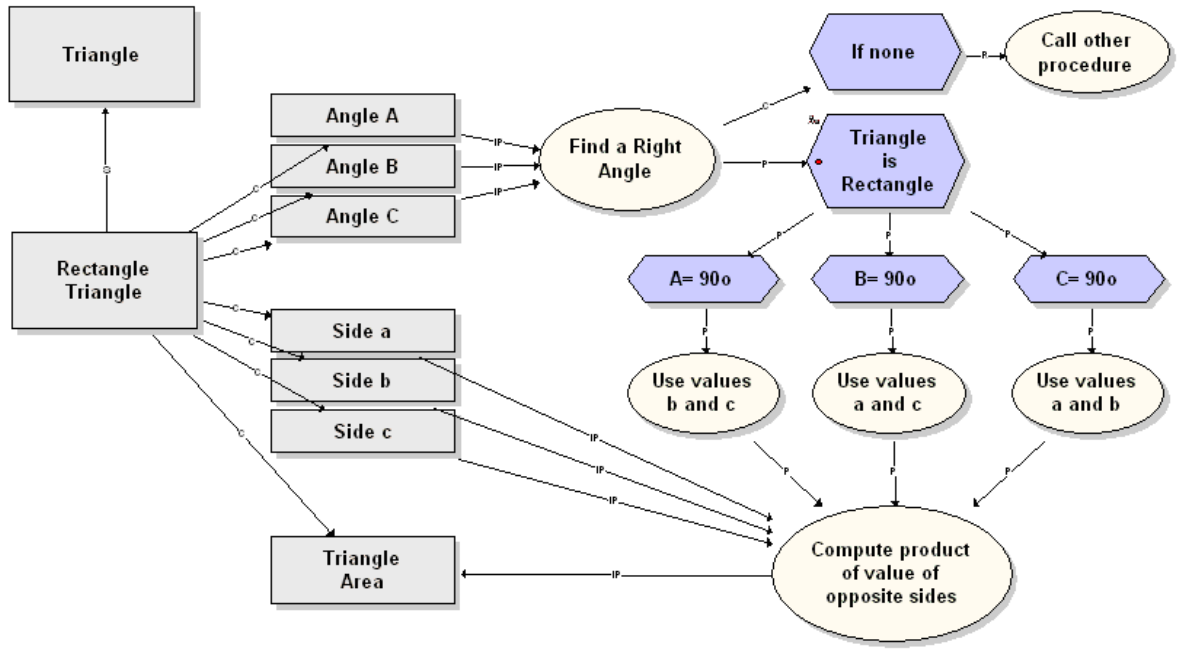
Schemas play a central role in knowledge construction and learning (Holoyak, 1991; Anderson et al., 1995). They defined perception as an active, constructive and selective process. They support memorization skills seen as processes to search, retrieve or create appropriate schemas to store new knowledge. They describe understanding as possible by the comparison of existing schema with new information. Globally, through all these processes, learning is seen as a schema transformation enacted by higher order processes, aiming at schema construction and reconstruction through interaction with the physical, personal or social world, instead of a simple transfer of information from one individual to another.

The distinction between conceptual and procedural schema has been accepted for a long time in cognitive science. More recently, a third category called "conditional or strategic schema" has been proposed (Paris, Lipson, & Wixson, 1983). These schemas have a component that specifies the context and the conditions to trigger a set of actions or procedures, or to assign values to the attributes of a concept. These categories map very well on the existing consensus in educational science.

The MOT Visual Modeling Language

We will now present briefly the syntax and semantic of the MOT visual modeling language, based on the notion of schema. Here, we could use graphs similar to UML object models to represent the attributes that describe a schema with different formats according to their type. In the MOT graphic language (Paquette, 1996, Paquette, 1999, Paquette, 2003), we have improved the readability and the user-friendliness of graphs by externalizing the internal attributes of a schema into other objects, with proper links to the original schema or object. For example, the link between the schemas "Triangle" and the "Rectangle Triangle" is shown explicitly using a specialization (S) link from the later to the former

Figure 1. A simple MOT model



concept. Links between the "Triangle" concept and its sides or angles attributes is externalized using a composition (C) link. The links from an input concept to a procedure and from a procedure to one of its products are both shown by an input/product (IP) link. The sequencing between actions (procedures) and/or conditions (principles) in a procedure is represented by a precedence (P) link. Finally, the relation between a principle and a concept that it constrains, or between a principle and a procedure that it controls, will be represented by a regulation link (R).

Using these links, this example on triangle concepts becomes the MOT model in figure 1 where relations between knowledge entities are transparent, mixing the types of entities and links.

Concepts (or classes of objects), *procedures* (or classes of actions) and *principles* (or classes of statements, properties or rules) are the primitive objects of the MOT graphical language. The type of the object is represented by geometrical figures as shown on figure 2, where each class or individual is represented by a name within the figure.

These objects are different types of schema whose attributes are all explicitly externalized and related to the schema using six kinds of typed links constrained by the following grammar rules:

1. All abstract knowledge units (concepts, procedures, principles) can be related by an instantiation **I** link to a set of facts representing individuals called respectively examples, traces and statements.
2. All abstract knowledge units can be specialized or generalized to other abstract knowledge using specialization **S** links.
3. All abstract knowledge units can be decomposed, using **C** links into other entities, generally of the same type.
4. Procedures and principles can be sequenced together using **P** links.
5. Concepts can be inputs to a procedure using an **IP** link to the procedure, or products of a procedure using an IP link from the procedure.
6. Principles can regulate, using an **R** link, any procedure to provide an "external" control structure, to constrain a concept or a set of concepts by a relation between them, or to regulate a set of other principles, for example to decide on conditions of their application.

Figure 3 summarizes these grammar rules of the MOT graphic language in the form of an abstracted graph where the entities represent types of MOT objects.

There are various possible semantic interpretations of these graphic symbols.

- **Concepts** can be object classes (country, clothing, vehicles, etc.), types of documents (forms, booklets, images, etc.), tool categories: (text editors, televisions, etc.), groups of people (doctors, Europeans, etc.), or event classes (floods, conferences, etc.).
- **Procedures** can be generic operations (add numbers, assemble an engine, etc.), tasks

Figure 2. Types of knowledge units in MOT

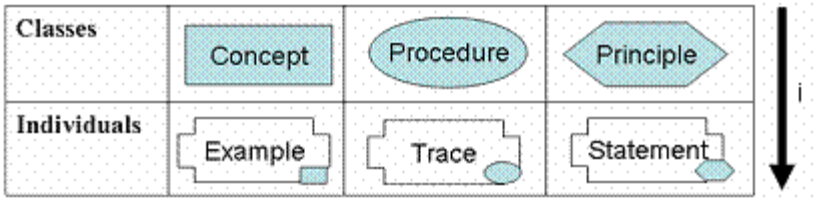
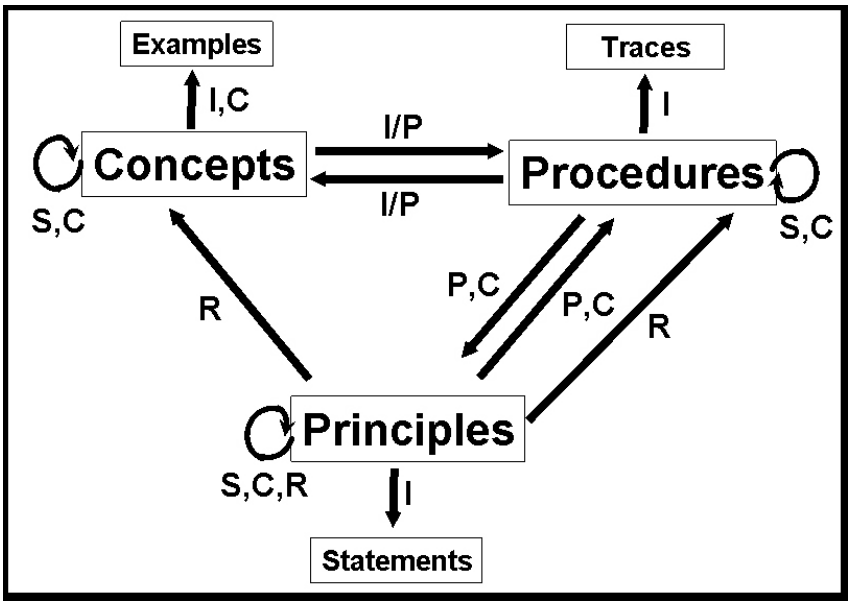


Figure 3. The MOT metamodel



categories (complete a report, supervise a production, etc.), activities (take an exam, teach a course, etc.), instructions (follow a recipe, assemble a device, etc.), or scenarios (of a film, of a meeting, of a learning module).

- **Principles** can state properties of objects (cars have four wheels), constraints on procedures (the tasks must be completed within 20 days), cause/effect relationships (if it rains more than 25 days, the crop will be in jeopardy), laws (any metal sufficiently heated will stretch out), theories (the laws of the market economy); rules of decision (advising on an investment), prescriptions (medicinal treatment, instructional design principles), etc.

The MOT+ Graphic Editor

With this set of primitive graphic symbols, it has been possible to build graphic models, from simple to complex representations of structured knowledge. For example, we can build representations equivalent to conceptual maps, flowcharts (iterative procedures) and decision trees, and also other types of models useful for educational modeling such as processes, methods and theories. All these types of models have been used in a number of projects since the first publication of the MOT editor in 1998, and also in the last 5 years with its extension to MOT+. Figure 4 presents examples of the main MISA visual models constructed with the MOT editor.

Figure 4a presents an example of a Knowledge model that describes part of the knowledge in the domain of artificial intelligence (AI) for an introductory Web-based course on that subject designed with MISA. Here ovals represent AI processes, rectangles represent AI concepts and hexagons represent AI principles.

Figure 4b presents a example of a *Pedagogical model* representing a learning scenario model for one the course modules where learning activities are represented as procedures (ovals) and learning resources as concept/object (rectangles).

Figure 4c presents an example of a *Media model* representing the structure of a Web site for the course. Concepts represent Web pages or page elements, ovals or circles represent hyperlinks,

Figure 4a. A knowledge model

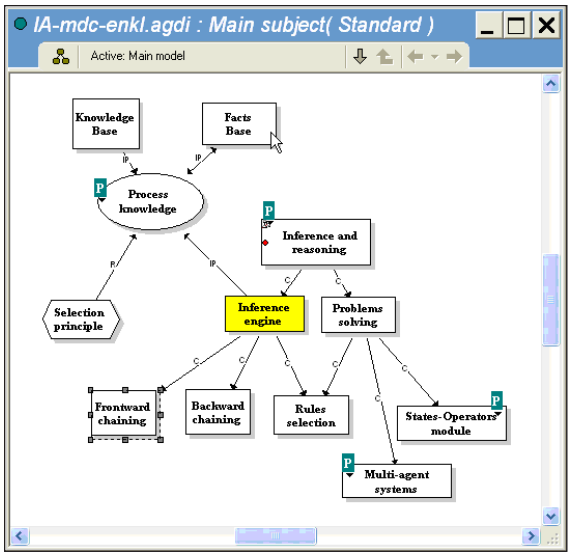


Figure 4b. A pedagogical model

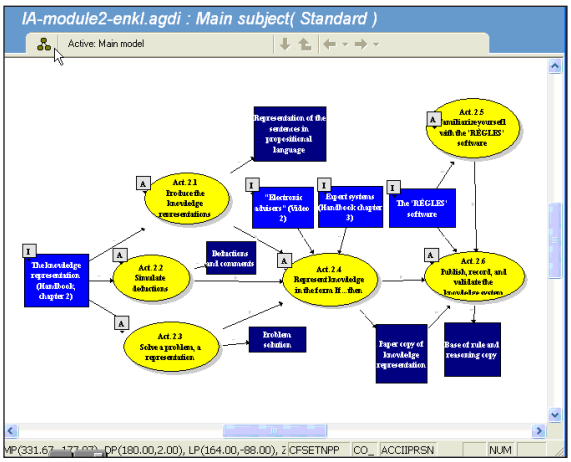


Figure 4c. A media model

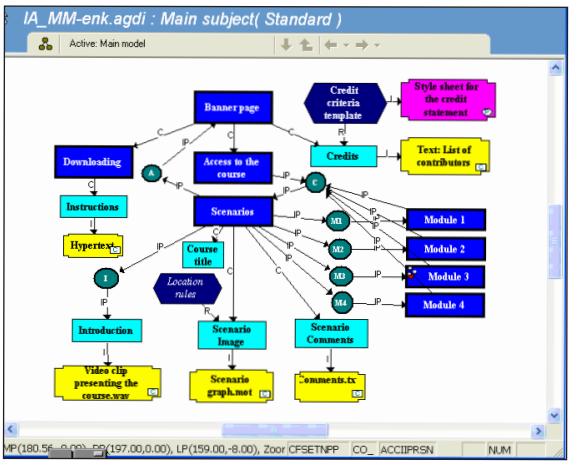
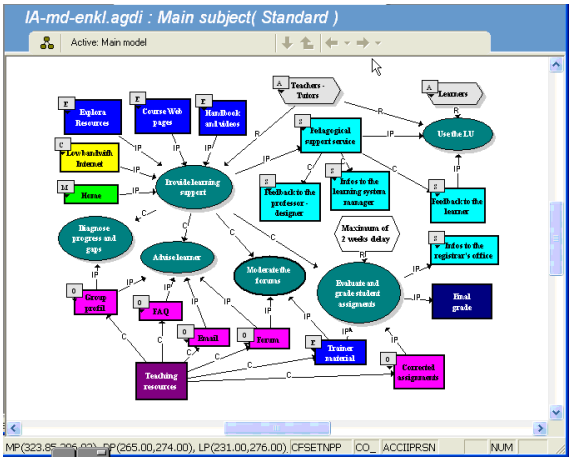


Figure 4d. A delivery model



as possible actions or procedures. Templates are represented by principles. Facts represent concrete object such as page elements with their actual texts, pictures or other resources.

Figure 4d presents an example of a *Delivery model* representing the course delivery process where actors are represented as control principles, acting on tasks represented as procedures, each having input and output resources.

This first version of the MOT editor has been extended to the MOT+ editor, a mature editor with advanced graphic editing capabilities (fonts, color, disposition on a page, etc.). Sub-models can be embedded at any depth and knowledge objects in each one can be displayed in a multi-layer mode. Models may be filtered in order to display only some types of knowledge objects or links. Sub-models from one model can be associated to objects in another model called a co-domain, which is very useful for example to assign knowledge to activities in a pedagogical model. Graphic objects can be associated to any type of document using the OLE standards such as a word document, slide presentation, Web page, spreadsheet or database file, which can be displayed by clicking on the graphic symbol. MOT+ has extensive export facilities to XML, HTML, Excel and other commonly used formats. In particular, the export to XML command provides the

possibility for graphic models to be processed by software agents respecting for example the IMS LD or OWL schemas.

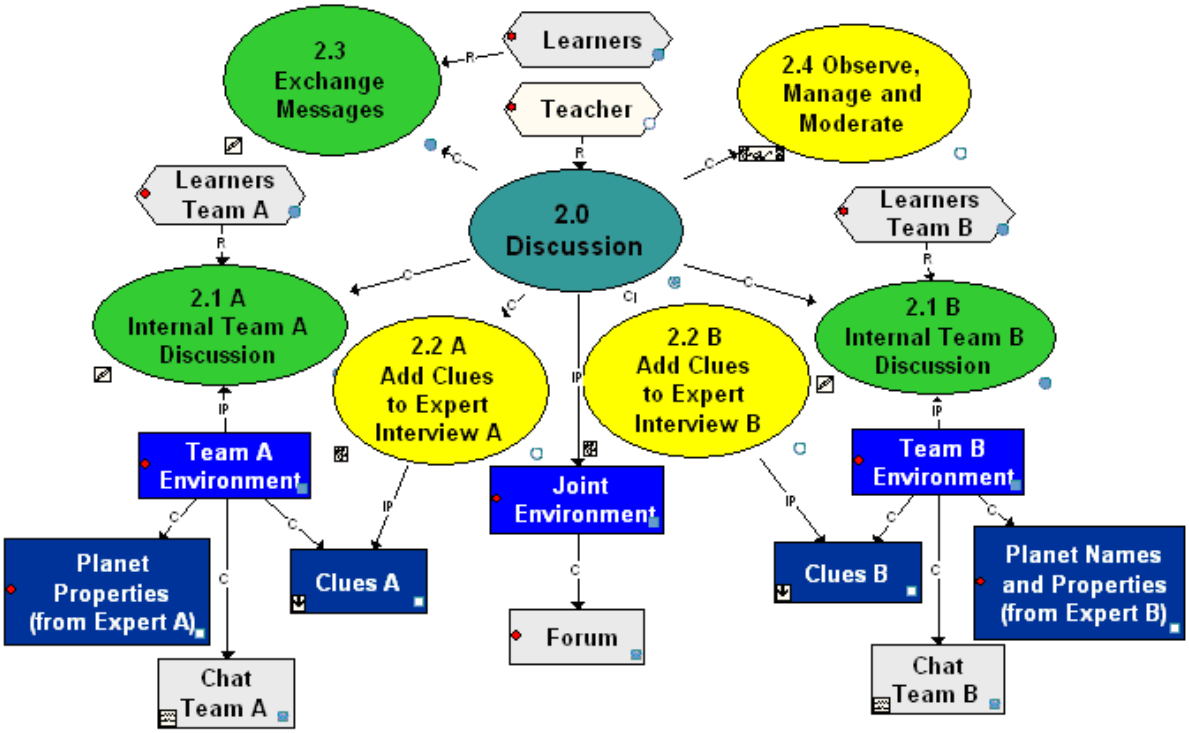
REPRESENTING MULTI-ACTOR WORKFLOWS AND LEARNING DESIGNS.

In the two following sections, we address the issues of the standardization of visual modeling languages, to promote the reusability of educational models and the interoperability between systems delivering learning environments. With the advent of an educational modeling standard specification like IMS-LD, we decided to develop a specialization of MOT+ to represent the IMS-LD concepts. During the eduSource and LORNET (ref) projects, we found that this specification was closely related to the MISA pedagogical model including some aspects of the MISA delivery model. This R&D is presented in section 3.1, the extension to a Web-based graphical editor is presented in section 3.2.

The MOT+LD Special Visual Language

IMS-LD provides a representation of the components of a learning environment in a standardized XML schema that can be executed by any compliant e-Learning platform. IMS-LD does not provide a visual language to build a learning environment specification. Initially, these had to be built using an XML editor or a form-based editor like RELOAD (2005). Also, IMS-LD is not an instructional design method to build such representations. It needs to be accompanied by any instructional design method, and MISA is more closely related than many other methods. Unfortunately, the MOT+ pedagogical models built in MISA are not executable on a variety of

Figure 5. An example of a MOT+LD learning design



platforms because they are not standardized. In fact, in the projects where we have used MISA, the specification was translated by hand, into the platform's activity editor, with some loss of information.

To address these problems, we first developed a graphic modeling editor for the IMS-LD specification (level A) and made it available as specialized editor in the MOT+ software. Many examples of learning designs have been produced by different groups using this editor. They can be found at the IDLD portal (2006). Figure 5 shows part of a simple example of a Unit of Learning (UoL) on solar astronomy presented recently at a workshop (Paquette & Léonard, 2006).

It shows an act and its activity structure containing various learning and support activities, all represented as MOT procedures (ovals). Since method, plays and acts, as the IMS-LD metaphor applies as concepts for an instructional structure, are also represented as procedures in other parts of the model. Each procedure type is indicated by a little label at the right lower corner of the ovals representing the procedures.

Similarly, roles are represented by different kinds of MOT principles (hexagons). Environments, learning objects, services and outcomes are represented by different kinds of MOT concepts (rectangles). Standard MOT links are used between these objects. C (is composed of), P (precede), R (regulate or govern) and I/P (input / product) links are sufficient to cover all the components of a standard IMS-LD level A learning design.

The MOT+LD editor is presented with some detail in (Paquette, Léonard, Lundgren-Cayrol, Mihaila & Gareau, 2006). It enables a designer to build graphically a compliant IMS-LD model. Afterwards, the graph is automatically validated and exported as an instance of the IMS-LD XML schema. This XML file can be read in form-based IMS-LD editors such as RELOAD (2005), if level B conditions and or level C notifications need to be specified. The XML can then be run by IMS-LD compliant players or platforms to deliver online learning sessions to their users.

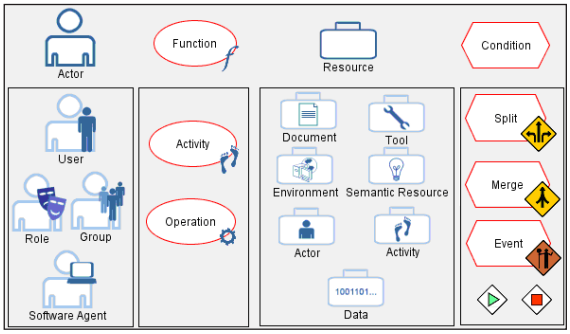
Paquette & Marino (2005) briefly discuss the strengths and weaknesses of the IMS-LD educational modeling specification. One weakness is the absence of knowledge representation, which is central to learning and knowledge management that we seek to support by the TELOS[3] system. We have proposed to improve that by the semantic annotation of the activities, resources and roles included in a learning design. A *semantic annotation* is a mapping from a subject matter ontology to the learning design that associates knowledge elements to the components of the design. This aspect will be developed in the following sections.

Extending the MOT+LD Editor

Another aspect of IMS-LD we need to improve is the control structure of the workflow, that is actually covered by level B and C specifications, where properties and conditions can be included in the design to alter the flow of activities, notify an actor or present a resource depending on previous actions or results stored in a user and group file or model. This aspect may not be that important in open learning environments where a total or large degree of liberty is left to the learner and facilitators, but for a business workflow in an organization, or to aggregate software components into larger resources, it is an important dimension.

To address that and provide a basis to build a function editor for the TELOS system, conceptual work on function maps has been defined as a central piece of the TELOS architecture (Rosca, 2005; Paquette, Rosca, Mihaila & Masmoudi, 2006). Moreover, a comparative analysis has been made between business workflows, IMS-LD learning designs and function maps (Marino et al., 2006), leading to the identification of 21 control situations for workflows encountered in software engineering literature (Correal & Marino, 2006).

Figure 6. Function Editor Symbols



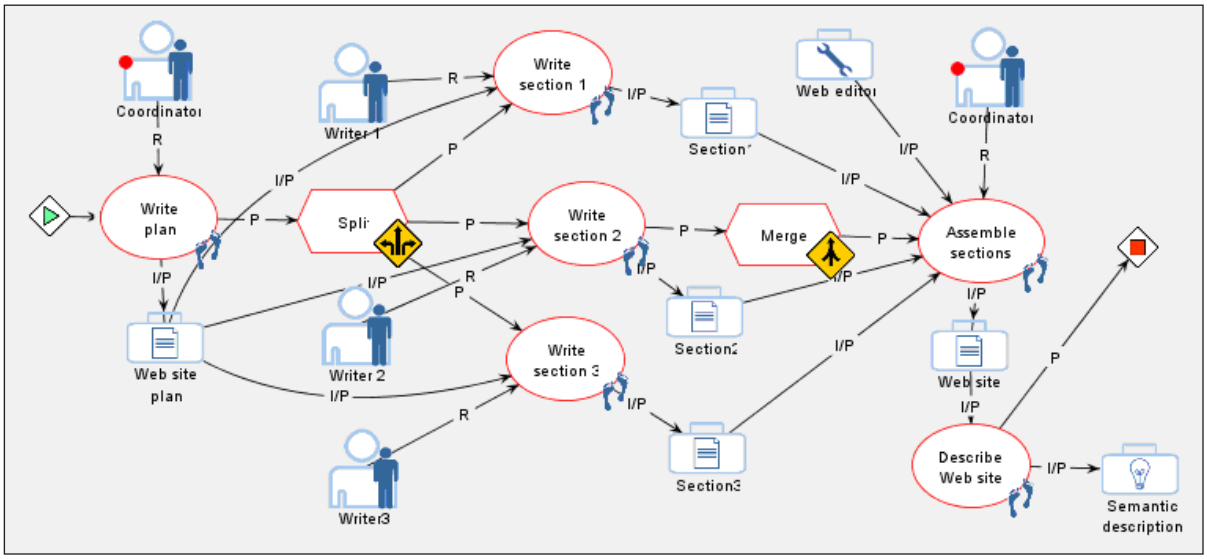
It was found that IMS-LD covers only some of these control situations, but probably the most useful ones for pedagogical design.

Based on this work and the actual MOT+LD editor, we are in the process of designing a new visual editor. The *Function Editor* aims both to generalize IMS-LD and to capture the main aspects of business workflows. The graphs produced by this editor will be used as executable interfaces for concrete actors to enact the activities and use the resources during delivery. It will also serve to orchestrate actors, activities and other resources, a fundamental principle built in to the TELOS system. A specialization of the function editor is being defined to cover all three levels of the IMS-LD specification.

The Function editor uses four kinds of MOT objects with subtypes taken from the TELOS technical ontology (Magnan & Paquette, 2006). These are shown on figure 6. *Concept* symbols represent all kinds of resources: documents, tools, semantic resources, environments, resource-actors, resource-activities and datatypes. *Procedure* symbols represent activities, including function models or commonly used operation templates to be embedded in other activities. Finally, *principles* are used both to represent different types of actors (as control agents) and control conditions. These two kinds of control entities are represented here by different symbols. The actor's symbols are active agents representing users, groups, roles or software agents that enact the activities using and producing resources as planned by the function model. Conditions are control element inserted within the basic flow to decide on the following activities that can be activated.

In figure 7, we see a combination of some of these symbols where a coordinator writes the plan of a document in activity-0. After that the figure shows a general split condition after activity-0. After that, activities 1, 2 and 3 are executed in parallel, controlled by the properties of the split

Figure 7. A simple function model



condition object. Later on, the flow of activities merges through the merge condition object before activity M+1 takes control. This activity will wait for some or all of the incoming flows to be activated before it is executed, again based on the properties of the merge condition object. The basic control flow is shown by P links and it is altered by R links. The data flow is shown by IP links.

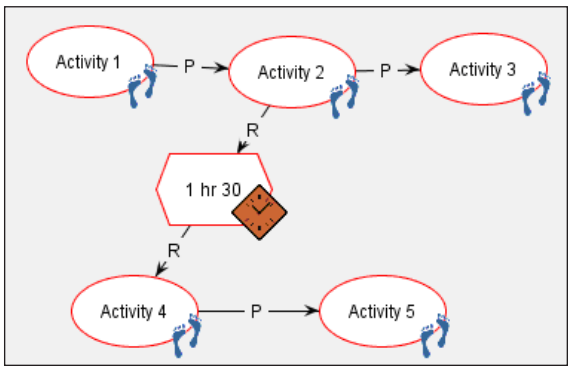
Figure 8 shows another kind of condition that alters the flow of execution. In activity-2, if the time-event condition is met, the flow of control will change. Depending on the type of the condition, the activity-4 will be shown or hidden. Activity-3 is still available. If activity-4 is shown and completed, then activity-5 can be performed.

Properties of the event condition symbol will provide the details on the condition and action parts of the control principle to provide the execution engine with a clear formal definition of the processing to take place.

In the Function Editor, we see a combination of a control flow and a data flow. The control flow is modeled using the MOT basic P and R links. P links indicates the basic sequence or flow of activities. R linked conditions identify which activities an event will trigger, thus altering the basic flow.

IP links from MOT serve to model the data flow, either from resources to activities where they are consulted, used or processed , or from activities to the resources they help produce. This

Figure 8. Event-based control



is why we need to distinguish between actors as active control entities and resource-actors that will serve as data providers or be products of an activity (e.g. a new person of software agent added in a system). A similar distinction is made for resource-activities that can be seen as resources to be transformed, for example by other activities creating or modifying their description.

C links from MOT may also be used to show the composition of an entity into other entities. A new unification, U link, is also necessary to guide the execution engine, when components are aggregated.

In TELOS, the function editor will enable engineers to combine resources into larger aggregates, technologist to built platform workflows for designers of learning or knowledge management environments, designers to build courses, work flows or learning /teaching scenarios.

MOT+OWL: A STANDARDIZED ONTOLOGY EDITOR

In section 1, we identified the pedagogical and the knowledge models as the most important ones. We now proceed with a second standardization task, that of the knowledge model. Any type of knowledge representation, including text-based narratives or informal graphic models, can be used to describe a domain of study. At the initial stage of design, the informal nature of an ontology representation is useful. The user's mind must be free to choose any representation that seems best suited for the educational project to be considered. Still, this very freedom does not facilitate the software processing of the representation.

Semi-formal modeling languages like MOT go part of the way in that direction. Unlike informal graphs built with any graphic editor, such as PowerPoint, the MOT graphic syntax is structured and has a general unambiguous semantic. Using the MOT editor, models can be exported in many formats, including a native XML schema. Using

this schema, software agents can perform different kinds of processing. Still, some ambiguity remains. In instructional engineering applications, we had to constrain the MOT graphic language even more to enable the delivery of learning scenarios in a digitized platform like Explor@-2 (Paquette, 2001). Even then, part of the transfer of the design to the delivery platform had to be done manually, to prevent enforcing unnatural graphic representations on the users.

The Ontology Web Language

To deliver computer-based learning environments, after a phase where informal graphic design has cleared up ideas, we need to move from informal or semi-formal graphs to formal computable graphic representations. Knowledge in a subject domain can be represented in many ways: taxonomies, thesauri, topic maps, conceptual graphs and ontologies.

We have selected to start with OWL-DL ontologies (see W3C, 2004) for a number or reasons. It is one of the three ontology Web languages that are part of the growing stack of World Wide Web consortium recommendations related to the Semantic Web. Of these three languages, OWL-DL has a wide expressivity and its foundation in descriptive logic guarantees its computational completeness and decidability. *Descriptive Logic* (Baader, Calvanese, Nardi, Patel-Schneider, 2003), is an important knowledge representation formalism unifying and giving a logical basis to the well known traditions of frame-based systems, semantic networks, object-oriented representations, semantic data models, and formal specification systems. It thus provides an interesting framework to represent knowledge for which a growing number of processing agents are built throughout the world.

OWL-DL provides a precise XML schema but no graphic representation per se. Some ontology editors like PROTÉGÉ (2006), provide some graphical views of the ontology, but the construction of an ontology is essentially form-based. Our goal was to provide a complete formal graphic representation of the OWL-DL that could combine the virtues of interactive construction with the computational capabilities of a formal graphic representation.

The MOT+OWL Visual Language

In the context of the MOT representation system, ontologies, in particular OWL-DL constructs, correspond to a category of models called theories. Ontologies can thus theoretically be modeled graphically using the MOT syntax. While doing this, we found out that while the MOT primitive objects and links were sufficient to represent ontologies expressed in OWL-DL, the graphs would become cumbersome unless new symbols were added. We have thus specialized the MOT language and graphic editor by adding sub-types for concepts, principles and facts and by adding new links.

Table 2 gives a few examples of the MOT+OWL graphic elements with their interpretation in descriptive logic and their correspondence to standard OWL-DL XML schema fragments. See (Paquette & Rogozan, 2006) for a complete description of the MOT+OWL graphic language.

Three types of MOT entities are sufficient to represent OWL-DL models. Concepts represent classes, principles represent properties and facts represent individuals. On these graphic entities, icons are added corresponding to axioms or principles stating a property of the class. We also added new special links to express things like equivalent "equi" or disjoint "disj" classes stating properties of two classes or two properties.

Figure 9. MOT standard equivalents

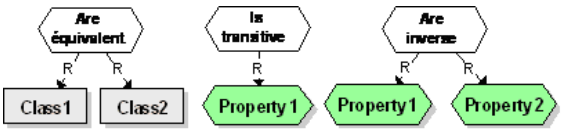
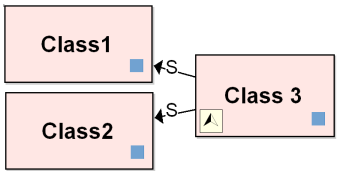
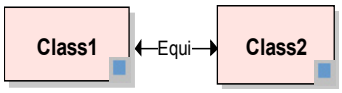
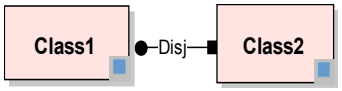
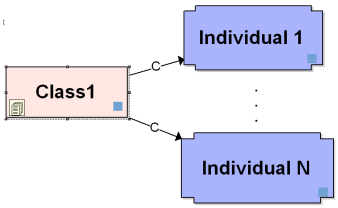


Table 2. OWL-DL equivalents

MOT+OWL graphic symbol	Description logic statements	OWL-DL XML-Schema segment
	Class intersection ∀x: Class3(x) ↔ Class1(x) ∧ Class2(x)	owl:Class> <owl:intersectionOf rdf:parseType="Collection"> *List of class descriptions* </owl:intersectionOf> </owl:Class>
	**Equivalent classes** ∀x: Class1(x) ↔ Class2(x)	<owl:Class rdf:about="#name_class1"> <equivalentClass rdf:resource="#name_class2"/> </owl:Class>
	**Disjoint classes** ∀x: Class1(x) ↔ ¬Class2(x)	<owl:Class rdf:about="#name_classe1"> <owl:disjointWith rdf:resource="#name_classe2"/> </owl:Class>
	Extension of a class ∀x: Class(x) ↔ (x = Ind 1) ∨...∨ (x= Ind N)	<owl:Class> <owl:oneOf rdf:parseType="Collection"> <owl:Thing rdf:about="#name_individual1"/> <owl:Thing rdf:about="#name_individual2"/> … <owl:Thing rdf:about="#nom_individualN"/> </owl:oneOf> </owl:Class>
	Functional property ∀x,∀y,∀z: Prop(x,y) ∧ Prop(x,z)) → y=z	<owl:FunctionalProperty rdf:about="#name_property" />
	Transitive property ∀x,∀y,∀z: Prop1(x,y) ∧ Prop1(y,z) → Prop1(x,z)	<owl:TransitiveProperty rdf:about="#name_property" />
	**Inverse properties** ∀x,∀y: Prop1(x,y) ↔ Prop2(y,x)	<owl:ObjectProperty rdf:ID="name_Property1"> <owl:inverseOf rdf:resource="#name_property2"/> </owl:ObjectProperty>

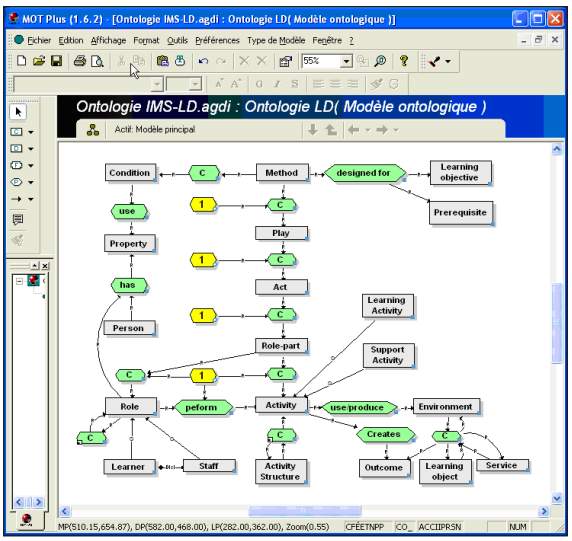
In the standard MOT syntax, these icons or special links would be expressed by principles with "R" links to classes or properties. For example, in the second and the two last examples of Table 2, the following standard graphs (figure 9) are equivalent, with the same precise OWL-DL interpretation as XML schema components. These would of course make the graphs more difficult for human interpretation.

Using a limited set of graphic symbols, we can formally describe any semi-formal MOT model that is amenable to a representation in descriptive logic. This is obviously the case for most conceptual models, laws and theory models. However, this is less evident in the case of procedural models, sometimes called task ontologies. Procedural and process/methods models are important for our purpose because learning environments are built around multi-actor processes.

Figure 10 presents a MOT+OWL visual graph that translates the conceptual structure of a learning design presented in the IMS-LD information model (2003). In the figure, "C" properties (green hexagons) are an abbreviation for "is-composed-of" which has the same meaning as the C link in standard MOT models, or the aggregation link in UML models.

This example illustrates the fact that functional relations between components of multi-actor processes such as a learning design can be

Figure 10. A simple task ontology for multi-actor scenarios[4]



represented by ontologies. Such ontologies have been used to test, for example, the conformance of particular learning designs to the IMS-LD XML schema (Amorim, Lama, & Sanchez, 2006), and to execute them in the context of an ontology-driven system.

Associating Knowledge & Competencies to Learning Designs

We have pointed out earlier the importance of associating knowledge and competencies (semantic annotation) to the components of a learning design. This is a key element of the MISA method. Actually, in IMS-LD, the only way to describe the knowledge needed to achieve the activities or that is present in the resources is to assign optional educational objectives and prerequisites, to the unit of learning as a whole and/or to all or some of the learning activities, but can not be added to express the level of competency for a support activity carried out by a teacher or tutor. Objectives and prerequisites correspond to entry and target competencies as used in the MISA method. They are essentially unstructured pieces of text composed according to the IMS RDCEO specification (IMS 2002).

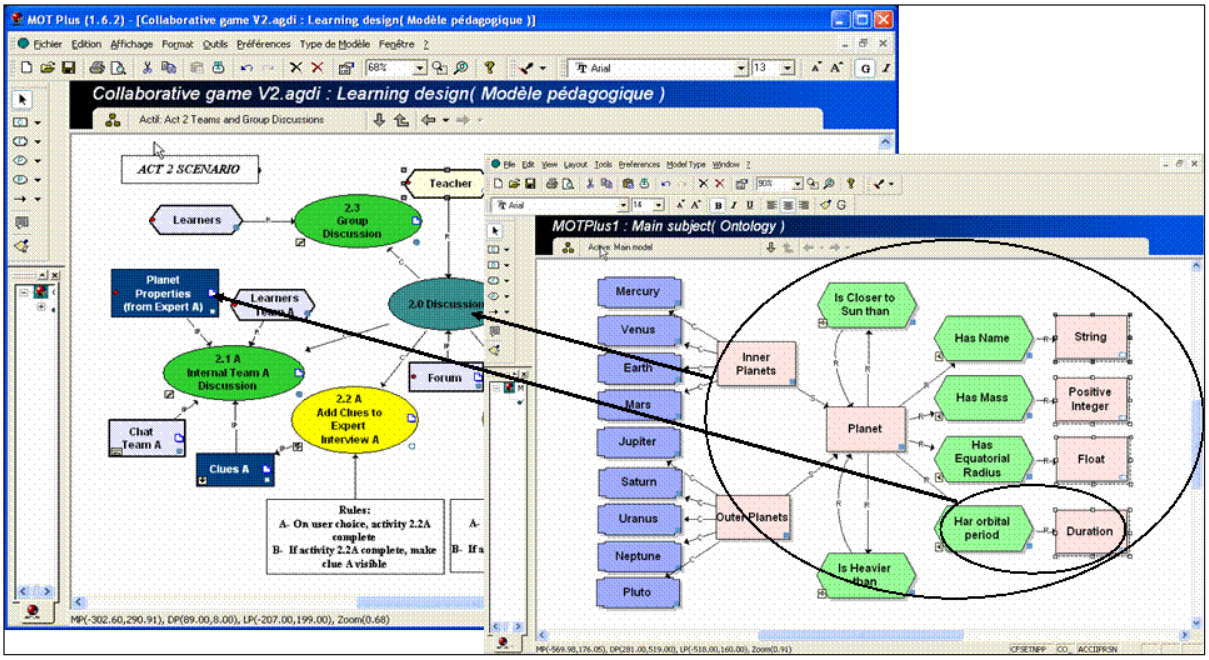
Unstructured texts are difficult to compare. Consistency checking between different levels of the LD structure cannot be supported computationally. Even at the same level of a learning design, for example within an act, no relations exist between the content of learning activities and of the input or outcome resources, and from these to the actors' competencies. In fact, in IMS-LD the knowledge represented in learning resources is not described at all, and the actor's knowledge and competencies are only indirectly defined by their participation in learning units or activities, if, and only if, educational objectives have been associated to the activities.

What we need first is a qualitative structural representation of knowledge and competencies associated to activities, resources and roles. This can be done using domain ontologies. As a first step, the MOT+ editor allows to show side by side a learning design, as well as associating it to a co-model, using the MOT+LD editor, and a domain knowledge ontology using the MOT+OWL editor. An example is shown in figure 6. The left hand window is the learning design presented earlier in figure 4. The right hand window presents part of domain ontology of the solar system (that was built before Pluto was declared a quasi-planet).

A *semantic annotation* is simply a mapping from the domain ontology to the learning design that associates knowledge elements (classes, properties and individuals of the ontology) to components of the learning design.

In figure 11, we see that data on the orbital period of planets in the solar system has been associated to a learning object in the design, which is a PowerPoint presenting this data to team A. This resource is an input to learning activity 2.1.A, but it is not the only input to this activity. There is also another resource (clues A) that gives additional information to team A, plus the chat between team members that will bring other clues to each participant. As a result, the sub-model of the ontology associated to activity 2.1A would logically correspond to the union of the sub-models of all input resources to the activity.

Figure 11. An example of ontology annotation of a learning design



Finally, the figure shows that most of the ontology model should be the subject of the discussion, since there is another team, team B that has more information to bring to the discussion using also information from input resources and in a team B chat. The larger sub-model is thus associated to the 2.0 activity structure.

This example shows how semantic annotation can help guide the construction of learning designs and to evaluate their coherence. By associating the right amount of knowledge to the different resources and activities, a designer can build a coherent design that will trigger collaboration between learners, or help a trainer decide on its intervention, or guide the actions of an intelligent tutoring system, and, in general support the evolution of the learners' competencies.

Desirable Properties in a Visual Educational Modeling Language

This chapter concludes with a discussion of the most important features and characteristics of a visual educational modeling language, which we think are the most useful and beneficial to the user.

Visual

The benefits of graphical cognitive modeling have been eloquently summarized by Ausubel (1968), Dansereau (1978), Novak (1993) and Jonassen, Beissner & Yacci (1993). Graphs illustrate relationships among components of complex phenomena. They uncover the complexity of actors' interactions and makes the most important parts stand out. They facilitate the communication about the reality studied. They favor the global comprehension of the phenomena under study. They help grasp the structure of related ideas by minimizing the use of ambiguous natural language texts. As an example, entity-relation graphs reduce ambiguity compared to a natural language description, but some remain on the interpretation of the terms written on the links or on the nodes. Ambiguity can be reduced further by the use of standardized typed objects and typed links.

User-Friendliness

Not all graphic modeling languages are user-friendly. A good counter-example is UML. The large number of models and symbols require considerable expertise and a steep learning time for the interpretation and for the construction of models. Furthermore, each type of model captures a different viewpoint of the information and it is impossible to mix them in the same graph to provide a global view of a subject domain. The representational system must be easy to use without technical or scientific mastery after a short period of initiation. Dansereau and Holley [39], have studied experimentally the use of different sets of graphic symbols by learners. Their results show that typed links are preferred by the majority of learners, as long as there are not two few nor two many links and they express sufficiently different meanings.

Generality

Generality means that the representation language should have the capacity to represent, with a relatively small number of object and link categories, all knowledge in very different subject domains, at various levels of granularity and precision. It should enable, to represent simple models such as a multiplication table, up to complex models such as multi-actor workflows, rule-based knowledge systems, methods and theories. It should also embed equivalent representations to commonly used graphs such as conceptual maps, semantic networks, flowcharts, decision trees or cause/effect diagrams.

Formalizable

The graphic language should be upward compatible from informal graphs, up to semi-formal and totally unambiguous formal models. At the informal level, an integrated representation framework facilitates the organization of thought and communication between humans about the knowledge which is exchanged, all along the evolution of the graphic representation model. Here the process is more important than the result. On the other end, the graphic language makes it possible to use more constrained elements to produce totally unambiguous descriptions that can be exported to set of symbols, such as an XML file, to be processed by computer agents. Here the model is more important than the process.

Declarative

Graphic language can be procedural or declarative. Procedural graphic languages have been built in the past; essentially extending flowcharts to promote graphical programming that would produce code directly. Our proposal is to use, as much as possible, a declarative graphic language, for a number of reasons. Firstly, it is easier for a person to declare the components of his/her knowledge than to describe the way it should be processed. In expert systems for example, the executive instructions are not wired-in the program, but externalized and made visible in a knowledge base on which a general inference engine proceeds. Secondly, the same model can be used for many different applications, not necessarily the one for which the processing has been planned in a procedural program. This is done by querying the model using an inference engine, in a Prolog-like manner. Thirdly, the processing knowledge itself can be given declaratively, so that higher order meta-knowledge, also can be singled-out. This idea is similar to structural analysis as proposed by Scandura (1973) and it is exactly the way we should see the relation between generic skills and domain knowledge in a competency, as meta-knowledge given declaratively, applied to domain knowledge, for example, rules for diagnosing a component-based system applied to different models describing a car, a software or a learning environment.

Standardized

Standardization is an important property to enlarge knowledge communication and use between persons or software agents. At the informal level, each model constructed by a person must be interpretable by another person. At the formal level, the communication capabilities extend to software agents. The move towards graphic versions of standards like IMS-LD for learning designs and OWL for ontologies adds wider communication capabilities between researchers and educators while at the same time adding formal non-ambiguous interpretation for machine processing.

Computability

Computability is a step beyond standardization. Not only can the graphic model receive a non-ambiguous formal representation that can be processed by computer agents, but this formal representation is complete (all conclusions are guaranteed to be computable) and decidable (all computations will finish in finite time). These considerations have motivated the construction of the MOT+OWL graphic language that is equivalent to the OWL-DL XML schema based on descriptive logic. OWL-DL ontologies are declarative, and standardized by the W3C.

CONCLUSION

This chapter has presented a 10-year effort to provide an educational visual language for applications that can span form informal support to idea generation, up to structured semi-formal graphs based on typed objects and links, and finally to graphic design on the formal conceptual and specification levels (MOT+LD, MOT+OWL).

In Botturi et al. (2006), the reader can find a classification of other visual languages, some of them being presented in other chapters of this handbook. According to this classification, MOT+ has the same properties as those of UML. It qualifies as a visual, layered, formal, conceptual and specification elaboration language, with multiple perspectives.

This corresponds to our initial goal of building a virtual language that is both user-friendly for designers (compared to UML) and still general and powerful enough to enable the design of the main components of a learning system, according to standard specifications. With the development of the new function editor based on MOT+ concepts, we can now go a step further and provide a visual scenario programming language that can be executed by an ontology-based engine to deliver usable learning environments to its users.

REFERENCES

Amorim R., Lama, M. & Sanchez, E. (2006). Using Ontologies to model and execute IMS Learning Design Documents. *The 6th IEE International Conference on Advanced Learning Technologies (ICALT-06)*, (pp.115-116). Kerkrade, The Netherlands.

Andersson, J.R. Corbett, A.T., Koeudinger, K.R. & Pelletier, R. (1995). Cognitive Tutors: Lessons learned. *The Journal of Learning Sciences, 4* (2), 167-207.

Ausubel, D. P. (1968). *Educational Psychology; A cognitive view*. New York, Rhinehart & Winston.

Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P. (Eds.). (2003). *The Description Logic Handbook*. Cambridge University Press.

Booch.G., Jacobson, J. & Rumbaugh, I. (1999). *The Unified Modeling Language User Guide*. USA: Addison-Wesley.

Botturi L., Derntl M., Boot, E. and Figl K. (2006) A Classification Framework for Educational Modeling Languages in Instructional Design. The 6th IEE International Conference on Advanced Learning Technologies (ICALT-06), (pp.1216-1220) Kerkrade, The Netherlands.

Bruner, J.S. (1973). *Beyond the information given*. New York: Norton.

Chen, P.P.S (1976).The Entity-Relationship model - toward a unified view of data. *ACM Transactions on Database Systems I, 1* (1), 9-36.

Correal. D., & Marino O. (2006). *Software Requirements Specification Document for General Purpose Function's Editor* (Technical Report V0.4), Montréal, Canada: Télé-université LICEF Research Centre, Montreal.

Dalziel, J.R. (2005). LAMS. Learning Activity Management System 2.0. Retrieved January 22, 2007 from http://wiki.oamsfoundation.org/display/lams/Home

Dansereau D.F. & Holley, C.D. (1982).Development and evaluation of a text mapping strategy. In A. Flammer & W Kintsch (Eds.), *Discourse Processing.* The Netherlands: North Holland.

Dansereau D.F. (1978).The development of a learning strategies curriculum. In H. F. O'Neil Jr., (Ed.), *Learning strategies*. New York: Academic Press. Davies.

Holoyak, K.J. (1991). Symbolic connectionism: Toward third generation-theories of expertise. In K.A. Ericsson & J. Smith (Eds.), *Toward a general theory of expertise: Prospects and Limits*. New York: Cambridge University Press.

IDLD (2006). Implementation and Deployment of the Learning Design Specification Portal. http://www.idld.org

IMS-LD (2003). Information Model, Best Practice and Implementation Guide, Binding document, Schemas. *IMS Learning Design*. Retrieved October 3, 2003, from http://www.imsglobal.org/learningdesign/index.cfm

Inhelder, B. & Piaget, J. (1958).*The growth of logical thinking from childhood to adolescence*. New York: Basic Books

Jonassen D.H., Beissner K., & Yacci M. (1993). *Structural Knowledge—Techniques for Representing, Conveying and Acquiring Structural Knowledge*. New Jersey: Laurence Earlbaum Associates

Magnan, F., & Paquette, G. (2006). TELOS: An ontology driven e-learning OS. In Weibelzahl, S., Cristea, A., (Eds.), Workshops held at the *Fourth International Conference on Adaptative Hypermedia and Adaptative Web-Based Systems 2006.* (pp. 131–139). Dublin, Ireland: National College of Ireland.

Marino, O., Casallas R., Villalobos J., Correal D. & Contamines, J. (2006).Bridging the Gap between e-learning modeling and delivery through the transformation of learnflows into workflows. In S. Pierre (Ed).*E-Learning Networked Environments and Architectures: A Knowledge Processing Perspective*. Springer-Verlag.

Merrill, M.D. (1994). *Principles of Instructional Design*. Educational Technology Publications, New Jersey: Englewood Cliffs.

Minski, M. (1975). A framework for representing knowledge. In P. H. Winston (Ed.), *The psychology of computer vision*. New York: McGraw-Hill.

Newell, A. & Simon, H. (1972). *Human problem solving.* NF: Englewood Cliffs.

Novak, J. D. (1993). How do we learn our lesson? Taking students through the process. *The Science Teacher, 60(3)*, 50-55.

OMG (2006). Business Process Modeling Notation (BPMM). Retrieved on July 24th, 2006 from http://www.bpmn.org/

Paquette, G. (1996).La modélisation par objets typés: une méthode de représentation pour les systèmes d'apprentissage et d'aide à la tâche. *Sciences et techniques éducatives, (4)*,9-42

Paquette, G. (1999). Meta-knowledge Representation for Learning Scenarios Engineering. In S. Lajoie et M. Vivet (Eds.), *Proceedings of AI-Ed'99 in AI and Education - Open learning environments,* IOS.

Paquette, G. (2001). Designing Virtual Learning Centers. In H. Adelsberger, B. Collis, J. Pawlowski (Eds.), *Handbook on Information Technologies for Education & Training* (pp. 249-272). The Netherlands: Springer-Verlag.

Paquette, G. (2002). *Modélisation des connaissances et des compétences, un langage graphique pour concevoir et apprendre.* Québec, Canada : Presses de l'Université du Québec.

Paquette, G. (2002). TeleLearning Systems Engineering – Towards a new ISD model. *Journal of Structural Learning 14*, 1-35.

Paquette, G. (2004). *Instructional Engineering for Network-Based Learning.* USA: Pfeiffer/Wiley Publishing Co.

Paquette, G., Crevier F. & Aubin, C. (1994).ID Knowledge in a Course Design Workbench. Educational Technology, (34) 9, 50-57.

Paquette, G., De la Teja, I., Léonard, M., Lundgren-Cayrol, K., & Marino, O. (2005). How to use an Instructional Engineering Method and a Modelling Tool (pp. 161-184), in R. Koper & C. Tattersall (Eds.). *Learning Design - A Handbook on Modeling and Delivering Networked Education and Training*, Springer-Verlag.

Paquette, G. & Léonard, M. (2006). The Educational Modeling of a Collaborative Game using MOT+LD. Proceedings from *the 6th IEE International Conference on Advanced Learning Technologies,* (pp.115-116), Kerkrade, The Netherlands.

Paquette G., Léonard M., Lundgren-Cayrol K., Mihaila S. & Gareau D. (2006). Learning Design based on Graphical Knowledge-Modeling. *Journal of Educational technology and Society ET&S*, *Special issue on Learning Design*, January 2006. (Also published in the Proceedings of the UNFOLD-PROLEARN Joint Workshop, Valkenburg, The Netherlands, September 2005 on Current Research on IMS Learning Design.)

Paquette, G. & Marino, O. (2005). Learning Objects, Collaborative Learning Designs and Knowledge Representation. *Technology, Instruction, Cognition and Learning, 3*, 85-108.

Paquette,G., Marino, O., De la Teja, I., Léonard, M., Lundgren-Cayrol, K., (2005). Delivery of Learning Design: the Explor@ System's Case (pp. 311-326). In R. Koper & C. Tattersall (Eds.). *Learning Design – A Handbook on Modelling and Delivering Networked Education and Training.* The Netherlands: Springer Verlag.

Paquette G., Marino, O., De la Teja, I., Lundgren-Cayrol, K., Léonard, M. & Contamines (2005). Implementation and Deployment of the IMS Learning Design Specification. *Canadian Journal of Learning Technologies,31(2)* (CJLT), http://www.cjlt.ca/

Paquette, G. & Rogozan, D. (2006). *Primitives de représentation OWL-DL - Correspondance avec le langage graphique MOT+OWL et le langage des prédicats du premier ordre*. TELOS documentation. Montreal, Québec: LICEF Research Center.

Paquette, G. & Rosca. I. (2004). An Ontology-based Referencing of Actors, Operations and Resources in eLearning Systems. SW-EL/2004 Workshop. The Netherlands: Eindhoven.

Paquette, G., Rosca. I, Mihaila S. & Masmoudi A. (2006 in press).TELOS, a Service-Oriented Framework to Support Learning and Knowledge Management. In S. Pierre (Ed).*E-Learning Networked Environments and Architectures: a*

Knowledge Processing Perspective. The Netherlands: Springer-Verlag

Paris S., Lipson M.Y., & Wixson K.K. (1983). Becoming a strategic reader. Contemporary Educational *Psychology, 8,* 293-311.

PROTÉGÉ (2006).Protégé Homepage: *Description and download available*. Retrieved July 24, 2006 from http://protege.stanford.edu/

RELOAD (2005).RELOAD Homepage: Editor and Player. Retrieved July 24, 2006 from http://www.reload.ac.uk/.

Romiszowski, A. J. (1981). *Designing Instructional Systems*. New York, USA: Kogan Page London/Nichols Publishing.

Rosca, I. (2005). *TELOS Conceptual Architecture*. (LORNET Technical Report: 0.5.).Canada: LICEF Research Centre, Télé-université.

Rumbaugh J., Blaha M., Premerlani W., Eddy F., & Lorensen W. (1991). *Object-Oriented Modelling and Design*. USA: Prentice Hall.

Scandura, J.M. (1973).*Structural Learning I: Theory and research*. London/New York: Gordon & Breach science Publishers.

Schreiber G., Wielinga B., & Breuker J. (1993). *KADS – A Principled Approach to Knowledge-based System Development*. San Diego, USA: Academic Press.

Sowa, J.F. (1984). *Conceptual Structures, Information Processing in Mind and Machine*. USA: Addison-Wesley Publishing Co.

Tennyson, R. & Rasch, M. (1988).Linking cognitive learning theory to instructional prescriptions. *Instructional Science, 17*, 369-385.

W3C (2004).OWL Overview Document. Retrieved February 10, 2004 from http://www.w3.org/TR/2004/REC-owl-features-20040210/

West, C. K., Farmer, J. A., & Wolff, P. M. (1991). *Instructional design: Implications from Cognitive Science.* Englewood Cliffs, NJ: Prentice Hall.

ENDNOTES

1 **MISA:** *Méthode d'ingénierie des systèmes d'apprentissage* is a French acronym meaning, "method for instructional systems engineering"

2 **ADISA:** *Atelier distribué d'ingénierie des systèmes d'apprentissage* is a French acronym meaning "distributed workbench for learning systems engineering"

3 **TELOS:** (TEleLearning Operating System) is a new system built within the LORNET project (www.lornet.org) to enable engineer and technologists to assemble eLearning and knowledge management platforms and environments.

4 On figure 10, principles with 1 express OWL cardinality axioms here meaning "at least one".